Energy Saving Through Intelligent Coordination Among Daily Used Fixed and Mobile Devices

Khan, R., & Khan, S. U. (2017). Energy Saving Through Intelligent Coordination Among Daily Used Fixed and Mobile Devices. *Sustainable Computing: Informatics and Systems*, *14*, 43-57. https://doi.org/10.1016/j.suscom.2017.03.003

Published in:
Sustainable Computing: Informatics and Systems

Document Version:
Peer reviewed version

Energy Saving Through Intelligent Coordination Among Daily Used Fixed and Mobile Devices

Rafiullah Khan[a,*], Sarmad Ullah Khan[b]

[a] *Queen's University Belfast, Belfast, United Kingdom, Email: rafiullah.khan@qub.ac.uk*
[b] *Politecnico Di Torino, 10129 Turin, Italy, Email: sarmad.khan@polito.it*

Abstract

Network end-user devices such as laptops and desktop PCs are often left powered ON 24/7 while they remain idle most of the time. The main reason behind this is maintaining network connectivity for remote access, VoIP, instant messaging and other Internet-based applications. The Network Connectivity Proxy (NCP) emerged as a quite promising strategy for significantly reducing network energy waste by allowing devices to sleep without losing their presence over Internet. It impersonates presence of devices during their sleeping periods at all layers of TCP/IP stack. The NCP can successfully proxy basic networking protocols (e.g., ARP, PING, DHCP etc) but faces challenges in proxying TCP sessions and proprietary closed-source applications with encrypted packets.

To overcome the limitations of NCP concept, this paper proposes a new strategy for reducing network energy waste through intelligent coordination among a user's devices (e.g., smartphone, laptop, desktop PC etc). In the proposed system, a user's daily used devices autonomously and seamlessly communicate with a centralized control server that decides which user device will maintain presence of applications at a given instant. To allow devices to sleep and wake them up whenever necessary, the proposed system also uses a light-weight Home Gateway Proxy (HGP) with very basic set of practically realizable features. The communication framework for HGP is designed to achieve auto-discovery, seamless networking and communication features. This paper also practically evaluates the proposed system and estimates energy saving for fixed as well as mobile network devices.

Keywords: Green Networking, Universal Plug & Play, Energy Efficiency, Power Measurement, Home Gateway Proxy.

1. Introduction

Several studies in literature have revealed that network end-user devices such as laptops and desktop PCs are left powered ON 24/7 in offices and homes even when idle [1]. A study by Lawrence Berkeley National Laboratory (LBNL) revealed that network infrastructure in US consumes 2% of total electricity which rises to 8% when considering servers and PCs as well [2]. Significant portion of electricity consumed by PCs is wasted as they are idle for at least two-third of the time everyday but still kept powered ON. The main reason investigated due to which people disable low power features, is the need of network connectivity for remote access, VoIP, Instant Messaging (IM) and other Internet based applications. It is worth to mention that Advanced Configuration and Power Interface (ACPI) defined low power states for compliant network devices as depicted in Fig. 1. The ACPI states are classified into four global states (G0-G3) which are sub-categorized into six sleep states (S0-S5). During idle periods, computers are in state S0 with all hardware components fully functional. However, computers can be put in a very low power standby state (i.e., S3 with typical power consumption of 2-4 W) during idle periods in order to reduce energy waste. It is also possible to wake-up the computer (i.e., S0) using remote wake-up technologies such as Wake-On-LAN (WOL) and Wake-on-Wireless LAN (WoWLAN) [3]. These ACPI low power states are rarely activated in most desktop computers in offices due to their inability to maintain network connectivity. Thus, huge energy savings can be achieved if computers sleep during idle periods [4].

The Network Connectivity Proxy (NCP) emerged as a quite promising strategy for significantly reducing network energy waste [5]. It allows network devices to sleep and impersonates their presence over the Internet [6]. The presence impersonation requires NCP to proxy sleeping devices at all layers of TCP/IP stack. To achieve this, the NCP generates/responds to packets on behalf of sleeping devices. The NCP needs to proxy basic networking protocols (e.g., ARP, PING, DHCP etc) as well as TCP sessions keep-alive messages and periodic application-specific heartbeat messages [7]. It wakes up a sleeping device (using WOL or WoWLAN technology) only when necessary e.g., remote access connection request [8]. Although, the NCP is a quite promising green networking strategy, it still faces several open issues and challenges especially related to proxying of TCP sessions and proprietary closed-source

*Corresponding author

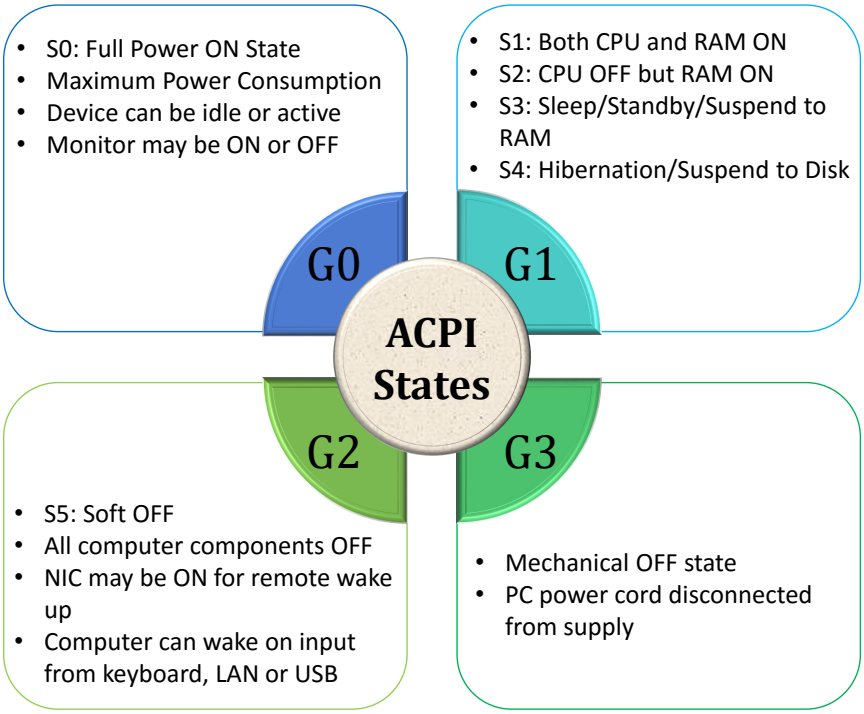


Figure 1: Overview of power states defined in ACPI specification for any compliant computer.

applications. The TCP proxying strategies in literature are either future oriented or propose changes to standard TCP/IP [6, 9, 10]. Further, application proxying strategies in literature are also either limited to only open-source applications or propose modifications to original applications [6, 11, 12].

This paper presents a new strategy that can effectively overcome the limitations of NCP concept. The proposed system reduces network energy waste through intelligent coordination among daily used fixed and mobile devices. Today, the smartphones and tablets are using the same operating system as the desktop PCs (e.g., Microsoft Windows 10, Ubuntu etc). This enables them to run the same applications and embed similar features as the desktop PCs. The proposed green networking strategy ensures that the applications are running on only single user device that is under active use at that particular instant. Thus, the applications will run on either desktop PC, laptop, tablet or smartphone at any given instant. For example, applications run on office computer during work hours and stop on all other user devices. If the user leaves office, applications start running on smartphone automatically and stop on all other user devices. This requires devices to seamless communicate with a centralized control server. It is worth to mention that applications on smartphone utilize hardware resources such as Wi-Fi, 3G/4G and built-in sensors which consume at-least 30% of its battery [13]. Thus, the smartphone can significantly improve its battery life in proposed system due to not running the applications 24/7.

The proposed system also uses a light-weight Home Gateway Proxy (HGP) to allow computers to sleep when idle (e.g., outside work hours). The HGP is subset of NCP and implements only a very basic set of practically realizable features. It only proxy basic networking protocols (i.e., ARP and PING) on behalf of sleeping computers and wake them up only when necessary (e.g., when remote desktop connection request is received). A communication framework is required to enable computers to communicate with HGP. This paper proposes the design of communication framework based on Universal Plug & Play (UPnP) technology that provides interesting features such as auto-discovery, seamless networking and communication. The proposed system significantly reduces energy waste of computers by allowing them to sleep and still accessible remotely whenever desirable. The proposed system is also useful for mobile devices. Although, energy savings from smartphones are negligible as being low power devices but can significantly improve their battery life.

The rest of the paper is organized as follows: Section 2 presents background and related work. It also highlights issues in previous proxying strategies. Section 3 presents the detailed design of proposed system. Section 4 presents the design of HGP. Section 5 presents implementation guidelines. Section 6 practically evaluates the proposed system and also estimates the energy savings. Finally, Section 7 concludes the paper.

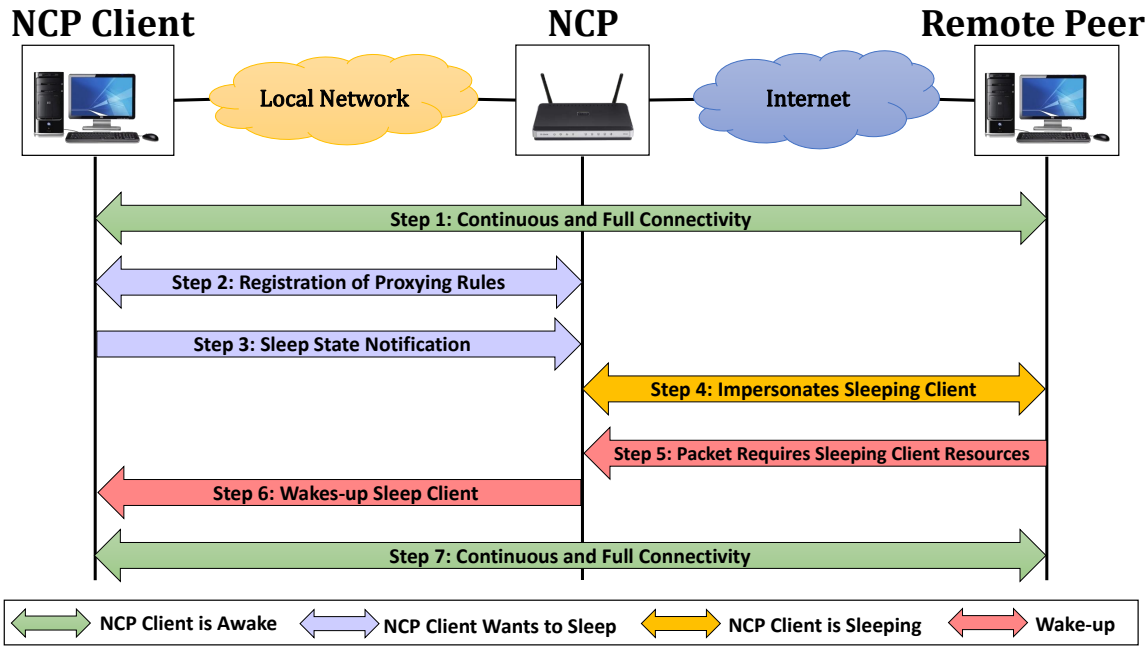


Figure 2: Overview of the NCP concept.

2. Motivation & Background Work

This section presents an overview of NCP concept and addresses related work from literature. Further, it also highlights issues and challenges in the NCP concept which are successfully overcome in our proposed collaborative proxying scheme.

2.1. Overview of NCP Concept

The objective of NCP is to hide the sleeping state of its client devices from remote peers over the Internet. Thus, it impersonates presence of sleeping client devices and makes them appear as online or connected to the Internet. This implies generating and responding to packets on their behalf. Presence impersonation requires proxying for sleeping devices at all layers of TCP/IP stack. The NCP needs to proxy basic networking protocol (e.g., ARP, PING, DHCP, IGMP, NetBIOS etc) as well as Internet based applications (e.g., VoIP, IM applications etc). To proxy applications, NCP needs to generate or respond to periodic application-specific heartbeat messages on behalf of sleeping devices. This also involves proxying of associated transport connections (i.e., TCP sessions). To understand application-specific heartbeat messages, the NCP developers need to have knowledge about the application source code.

Fig. 2 presents an overview of NCP concept consisting of the following steps: *Step 1:* The NCP client is awake and maintains its full connectivity with remote peer over the Internet. *Step 2:* The NCP client registers with NCP and also registers required proxying behavior e.g., the applications and protocols to proxy. *Step 3:* The NCP client becomes idle and notifies its power state to NCP just before entering into sleep state. *Step 4:* The NCP client is sleeping and NCP impersonates its presence with remote peer over the Internet. *Step 5:* The NCP received a packet that requires sleeping client device resources e.g., remote desktop connection. *Step 6:* The NCP wakes up its client device by using WOL or WoWLAN technology. *Step 7:* The NCP client has woken up, retrieved current proxying state from NCP and resumed its full network connectivity over the Internet. The operations in Fig. 2 are highlighted with different colors: green (i.e., NCP client is awake), blue (i.e., NCP client wants to sleep and interacts with NCP), orange (i.e., NCP is proxying its sleeping client device) and red (i.e., the wake-up process of NCP client).

2.2. Literature Review

Several researchers have investigated the NCP concept, although with different names. Its basic requirements have been analyzed in [5, 10]. Authors in [3] investigated different types of NCPs. Based on the design and deployment in local network, authors have also identified challenges and proposed suitable solutions. ECMA [14] standardized a reference proxying framework for sleeping devices. The standard specifies mandatory and optional requirements. Further, different proxying modes and deployment options in the local network have been identified. Authors in [1, 15] motivated the NCP concept by estimating the potential energy savings. Authors in [1] also investigated what applications and protocols need proxying and what can be ignored by performing detailed network traffic analysis in home and office environments. Most literature works addressed proxying of a specific open-source application or network protocol such as XChat and kaduChat

[4], Gnutella [16], Jabber clients [17], UPnP protocol [18]. Few researchers have investigated a rather more complete proxying framework using different hardware platforms [6, 11, 19].

The NCP concept has successfully addressed proxying of basic networking protocol (e.g., ARP, PING, DHCP, IGMP, NetBIOS) and source-source applications. However, proxying of TCP sessions and proprietary closed-source applications is quite challenging. Authors in [9] proposed green TCP concept by introducing a new header field 'Connection Sleep' inside TCP packets. This modification to standard TCP/IP informs remote peer about the device power state transitions. Authors in [10] proposed a split-TCP feature by introducing a new 'shim' layer between application and socket interface. The shim layer communicates with shim layer on remote peer to freeze and resume a TCP connection. Another strategy to preserve TCP connections on behalf of sleeping devices is the use of Srelay SOCKS service [5]. It relays every TCP connection between both communicating peers. Authors in [6, 20] proposed TCP session migration feature. The NCP client freezes its TCP session before sleeping and transfers the state to NCP. The TCP session is migrated back to NCP client when it wakes up. The TCP session migration uses a feature in Linux kernel by putting TCP socket in repair state and adjusting its parameters such as sequence and acknowledgment numbers. The TCP session migration feature needs to be implemented by all Internet-based applications. It can be observed that all TCP proxying strategies are either future oriented or propose modifications to standard TCP/IP and applications. Further, the strategies face scalability issues and real-time performance might be degraded.

To proxy applications, the NCP needs to generate/respond to application-specific periodic heartbeat messages. Authors in [11, 19] proposed the idea of using application stubs. The application stub is a light-weight version of original application re-written from its source code. Thus, writing stubs is a very challenging process and developers need to understand different programming languages (i.e., applications are normally developed in different languages). Further the strategy can be used only for open source applications. Authors in [4] proposed application specific routines and tested for xChat and kaduChat applications. The NCP client device continuously monitors application heartbeat messages based on transport protocol and port number. The last heartbeat message is provided to NCP before going to sleep state. The NCP then modifies heartbeat message in order to generate/respond to future heartbeat messages on behalf of sleeping client device. This strategy is useful if payloads are unencrypted and predictable (e.g., payload value is a counter, random value etc). Authors in [6] proposed application freeze and resume features. An application on NCP client device freezes operations and transfers the state to NCP along with TCP socket details. The NCP resumes operations when client device enters into sleep state. The application state is transferred back to NCP client after it wakes up. This strategy also requires heartbeat message payload to be unencrypted and predictable (e.g., payload value is a counter, random value etc). Further, it requires modifications to applications. All applications need to implement freeze and resume features. It can be observed that all application proxying strategies are limited to open-source applications or require modifications to applications to include new features.

Proxying techniques are also useful for mobile devices in order to improve their battery life. Authors in [13, 21] analyzed the applicability of NCP concept for mobile devices. Mainly issues and challenges were analyzed due to device mobility. Additionally, the NCP also faces the same challenges as for fixed network devices (e.g., proxying of TCP sessions and proprietary closed-source applications). Authors in [22] used proxying approach for dynamic power aware scheduling. The proxy ensures to transmit data in burst by splitting the connection between both communicating peers. The burst transmission slightly reduces energy consumption by allowing network interfaces to sleep between bursts. Authors in [23] and [24] analyzed similar strategy for BitTorrent and delay sensitive applications (e.g., YouTube, NetFlix, etc), respectively. Authors in [25] presented the design of 'scepter' which is a stateful proxy for reducing control and data bits during transmission between both peers. However, energy saving potential of data compression or size reduction is much lower compared to the NCP concept.

2.3. Issues & Challenges in NCP Concept

Although NCP seems to be quite optimistic green networking strategy, it faces several unresolved issues and challenges:

1. The application proxying strategies for NCP are only limited to open-source applications or propose changes to original applications [6, 11]. It is worth to mention that majority of daily used applications (e.g., skype, viber, VoIP clients etc) are proprietary closed-source which cannot be proxied by NCP.
2. Even to implement proxying for open-source applications, detailed knowledge of their source code and programming languages is required. This makes the task quite challenging due to ever increasing number of applications.
3. It is very challenging for NCP to proxy applications with encrypted payloads. For each application, the NCP needs to know its security policies and keying material (e.g., encryption and signature algorithms, keys, keys validities etc). Such information is normally not shared by application developers especially in case of proprietary closed-source applications. Proxying task becomes even much more challenging for applications which use end-to-end encryption (e.g., WhatsApp).

4. Proxying of open TCP sessions on behalf of sleeping devices is also very challenging. Proposed strategies in literature either modify the standard TCP/IP or face scalability and performance limitations [5, 6, 10].
5. The NCP coverage is only limited to local network devices. Thus, if a mobile device e.g., laptop leaves the local network, it cannot be proxied or woken-up.
6. Proxying of mobile devices is very challenging for NCP as it needs to track the devices mobility and changing IP addresses [21]. Further, NAT/Firewall may prevent NCP communication with mobile devices when they travel a different network.

All these issues don't exist in the implementation of our proposed system which makes it practically realizable. The proposed system does not proxy applications but instead runs them on any active user device based on priorities or specified settings. Further, there is no need to proxy TCP sessions in the proposed system. Since, the proposed system uses a centralized control server, the mobility of devices does not cause any issues. To allow remote wake-up of devices as in NCP, the proposed system uses a light-weight HGP. The HGP wakes up a sleeping computer when a new connection request is received (e.g., a user trying to access his sleeping computer at home from office or vice versa using remote desktop). Thus, all the functionalities in proposed system are practically realizable with achievable energy savings equivalent or higher than the NCP concept.

3. Proposed System: Intelligent Collaborative Proxying Scheme

The proposed system consists of two parts: (i) intelligent coordination among daily used fixed and mobile devices for maintaining connectivity for applications (addressed in this section) and (ii) HGP for maintaining connectivity for remote desktop/access (addressed in Section 4).

3.1. Overview

Today, people are using multiple computing devices in their daily life including smartphones, tablets, laptops and desktop computers. The modern laptops and desktop computers are very powerful and equipped with 3+ GHz processors, up to 16 GB memory and 8 GB disk space. Till few years ago, the smartphones were very weak computing devices with very low processing power and available memory. However, due to advancement in technology, today smartphones are very powerful with quad-core and octa-core processors, more than 2 GB memory and storage space of 128 GB. Further, smartphones today can run the same applications as well as the same OSes as desktop computers such as Microsoft Windows 10 and Ubuntu. Ideally, a modern smartphone can do everything that a desktop computer can do.

The smartphones got increasing popularity and millions of people are using an Internet-connected smartphone in daily life. Thus, why not maintain presence of applications on a smartphone during the time periods when the user is not using his desktop computer. Thus, the applications will run on a single user device that is under active use at that specific moment. The original NCP concept was proposed with two objectives: (i) proxy basic network presence of sleeping computers and wake-them up only when necessary (e.g., remote desktop connection request) and (ii) proxy the presence of applications on behalf of sleeping computers. In our proposed system, the first objective is achieved with a light-weight HGP with very basic set of practically realizable features. The design and features of HGP are addressed in Section 4. The second objective of NCP in our proposed system is achieved through intelligent coordination among daily used devices by relying on a centralized control server. Applications on smartphone normally run 24/7 and consume significant portion of battery [13]. The proposed system improves battery life of smartphone as it will run applications only when no other user device is running them. Further, huge energy savings can be achieved from desktop computers as they can sleep during idle periods and their basic network presence being maintained by the HGP.

3.2. Design of Proposed System

The proposed system uses a central Application Coordinating/Controlling Unit (ACU) with which all user devices communicate. The ACU is a global entity running anywhere in the world and manages user accounts and their registered devices. It guarantees to run applications on either a smartphone or tablet or laptop or desktop PC or any other device currently under active use. The user first needs to create an account with ACU and register all of his devices and applications. Fig. 3 depicts the generic scenario for a single user having different devices. This scenario assumes that the user has three devices: one desktop computer in office, one laptop at home and one smartphone which is always with the user where ever he moves. All of these three devices communicate with ACU which instructs them to run or not to run the applications. E.g., when the user is in office, the ACU informs office computer to run all the applications (such as skype, viber, VoIP clients and others) while informs home laptop and smartphone to stop the applications. During this time, the home computer/laptop can sleep and can be woken up whenever necessary by HGP running on the gateway device. Also the smartphone during this time will have lower utilization of CPU, memory, Wi-Fi, 3G/4G, built-in sensors etc due to stopping all the applications. The smartphone and home laptop periodically check with the ACU if they need to resume all the applications. When the user leaves office, his office computer enters into sleep mode and HGP in office network will manage its availability for remote access. The office computer is using a kernel module that tracks changes in its power state transitions. As soon as the office computer receives sleep instruction and detected by kernel module, it immediately notifies the

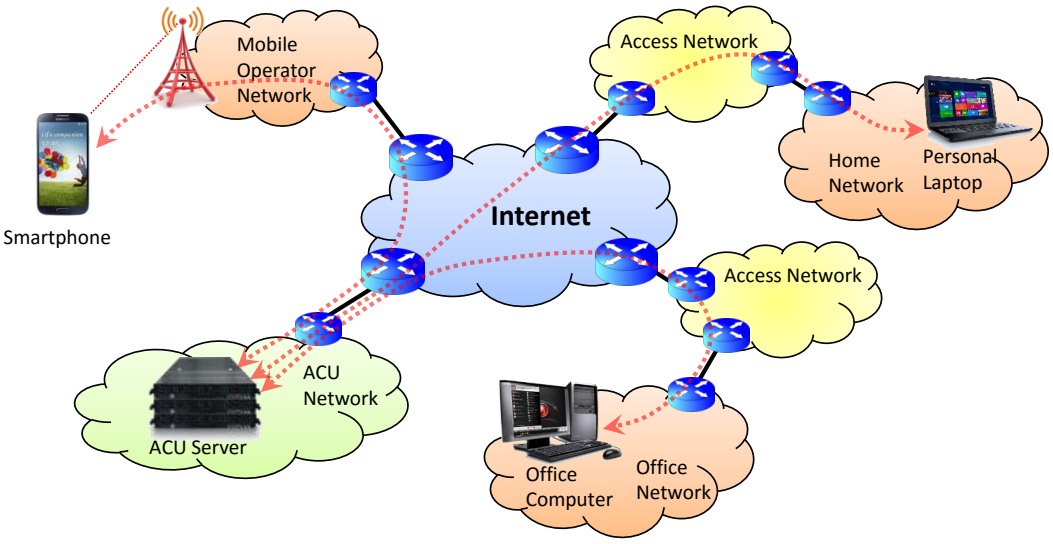


Figure 3: The overview of proposed system. A user has three different devices: a laptop at home, a desktop PC in office and a smartphone. The ACU server adopts devices priorities or user-specified settings and runs applications on only a single device at a given moment.

ACU. Meanwhile, the ACU informs smartphone to run all the applications. Likewise office computer, the home computer/laptop also uses kernel module with the same objectives. This way, the applications run on only single user device based on his location at the given moment.

Fig. 4 depicts the generic architectural design of proposed system. The ACU is a global entity and may offer its services to millions of users around the world, each with multiple devices. Fig. 4 assumes a total of N users who created account with the ACU server. During account registration, the user specifies all of his devices and provides custom settings or assigns priorities to devices. The user in his account can also mention the applications that he will be using on his devices. Now based on the custom settings or priorities, the ACU server runs applications on only single user device at the given moment. Fig. 4 also depicts different user devices that communicate with the ACU server. Each device runs an ACU client that enables the device to create an account with the ACU server or add itself to an existing account, add/remove applications, update priorities etc. The ACU client also includes an Application Controller who is responsible to run or stop applications on the device based on the instructions received from ACU server.

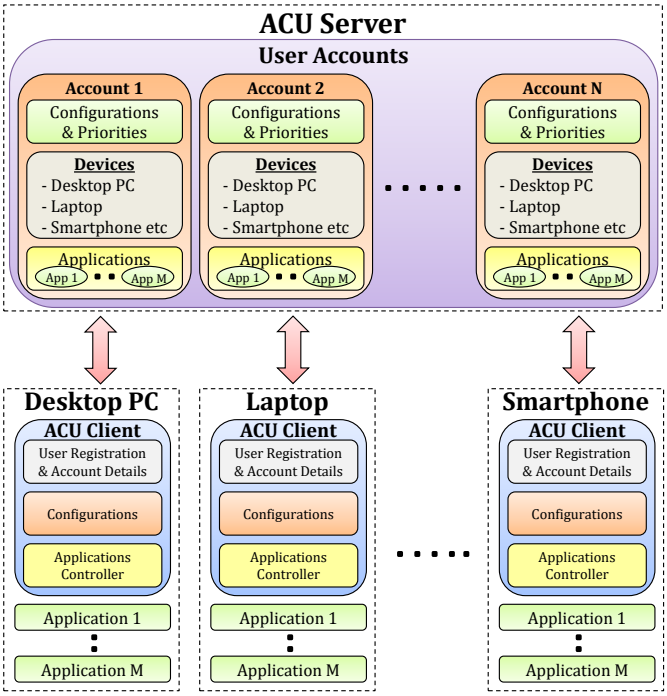


Figure 4: The generic architectural design of proposed system.

To have better understanding of the ACU client functionalities, Fig. 5 depicts a basic flow chart. Once the ACU client software is started, it registers or logins at the ACU server. Meanwhile, it also asks the ACU server whether it needs to start the applications or not. The ACU server may inform the device not to start applications based on the device priority or user-specified settings. The ACU client will periodically send a heartbeat message to announce its presence to the ACU server and also to know if its the time to start applications. If there is no high priority registered device at the ACU server at the given moment or high priority device de-registers with the ACU server, it informs the ACU client to start the applications. After starting applications, the ACU client continues sending periodic heartbeat messages to the ACU server to announce its presence and to check if it needs to keep the applications running. If in meanwhile the ACU server discovered another high priority device, it will instruct the ACU client to stop the applications.

Fig. 6 shows the basic flow chart for ACU server to have more clear understanding of its functionalities. The ACU

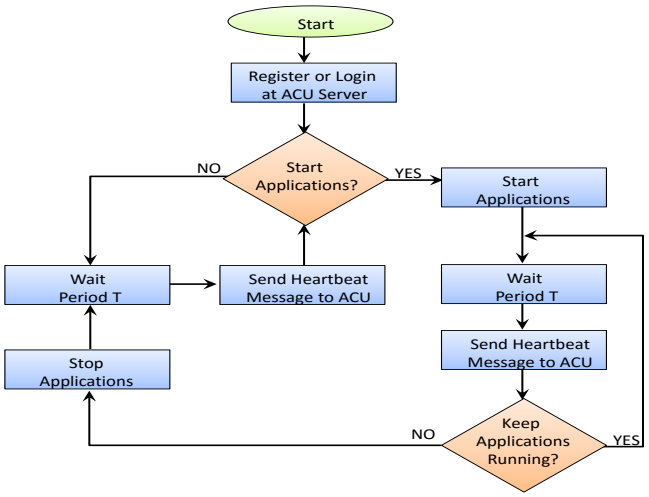


Figure 5: The ACU client flow chart.

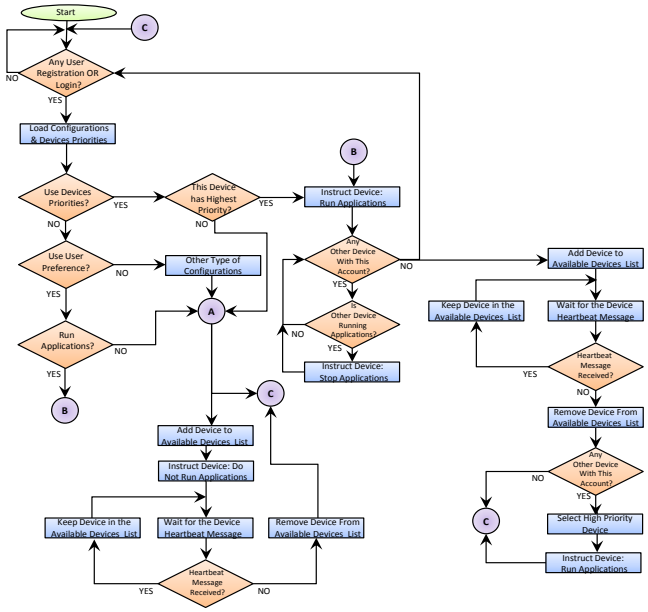


Figure 6: The ACU server flow chart.

server continuously checks for new account registrations or logins to the previously created accounts. Once a device login, the ACU server loads user configurations and device priorities for that specific account. It checks if this account has requested controlling of applications based on device priority, user preference or other type of settings. If the decision is requested based on priority, it checks if this device has the highest priority. If the device has low priority, the ACU server will instruct it not to run the applications and vice versa. If the decision is requested based on user preferences, the ACU server checks if this device should run applications or not. Meanwhile, the ACU server also checks all other registered devices with this account to know if any other user device is running applications. If yes, it will instruct the other device to stop the applications. The ACU server also checks if a device periodic heartbeat message is received on time, otherwise deletes the device from registered devices list and instructs the next suitable device to run the applications.

3.3. Application Control Logics

It is worth to mention that Fig. 6 shows very basic flowchart for the ACU server based on the devices priorities and user preferences. However, the ACU server could be much more complex by implementing different application control logics. The following are possible application control logics for ACU server:

3.3.1. Based on Device Priority

In priority based application control, the ACU server runs applications on the highest priority registered device and stops on all low priority devices. For example, smartphone has lower priority compared to tablet, tablet has lower priority compared to laptop and laptop has lower priority compared to desktop computer. Thus, the ACU server first checks if desktop computer is registered and awake. It will only instruct smartphone to run applications if no other device is registered or other devices are in sleep state.

The ACU server may also provide a feature to transfer control of applications at any time to low priority device based on user request even if the high priority device is available.

3.3.2. Based on Pre-specified Time Periods

In this application control logic, the ACU server runs applications on devices based on time of the day. For example, office computer runs applications from 8:00 am until 6:00 pm. The home computer runs applications from 8:00 pm until 10:00 pm. The smartphone runs applications at any other time when office and home computers are not running them. The user may also specify that no device should run applications during his sleeping time from 10:00 pm until 8:00 am.

3.3.3. Mixed User Configurations

The ACU server may control applications based on mixed user-specified configurations. The user can specify to run some applications on one device and some on the other device. For example, the user has total 5 applications: A, B, C, D and E. Smartphone will run applications C and D. Office computer will run applications A, B and E. The home computer will run all applications.

3.4. Communication Framework

It can be observed in Fig. 3 that a communication framework is required for communication between user devices and ACU server. In the proposed system, this communication framework is based on HTTP protocol that works in the client-server fashion. The choice of HTTP is

due to its structured meta-data and open flow (at-least in outbound direction) through Firewalls. The HTTP protocol in its original form has no built-in security and faces security threats due to transmission of unencrypted packets over the Internet. To protect HTTP communication against cyber attacks, the Group Domain of Interpretation (GDOI) [26] based security mechanism is used. The GDOI provides enhanced protection against traffic manipulation attacks due to its periodic security policies and keying material refreshment mechanism.

The GDOI mechanism consists of a Group Controller and Key Server (GCKS) and Group Members. The GCKS provides security policies and keying material to group members which they use then for communication among each other. In the proposed system in Fig. 3, The ACU server also plays the role of GCKS and provides security policies and keying material to registered user devices. The first step in GDOI security mechanism is secure authentication of user devices with ACU server. This authentication phase is based on Diffie Hellman exchange [26]. Once devices successfully authenticate, the ACU server (i.e., GCKS) provides them new updated security policies and keying material on periodic basis. The security policies contain information about the expiry of keying material as well as encryption and signature algorithms to use. Current implementations support AES-128-GCM, AES-256-GCM and AES-256-CBC encryption algorithms and HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 and AES-GMAC-128 signature algorithms. Thus, encryption and signature algorithms for protecting HTTP communication are changed on periodic basis and decided by the ACU server (i.e., GCKS).

The GDOI provides strong protection to HTTP against cyber attacks targeting integrity and confidentiality of communication. The authentication/access and traffic manipulation attacks (e.g., man-in-the-middle) are prevented due to encryption. The periodic security policies and keying material refreshment mechanism provides protection against replay and cryptanalysis attacks. The GDOI group-based security also provides protection to user privacies in proposed system. As depicted in Fig. 7, every user has different security policies and keying material. However, all devices belonging to a single user have same security policies and keying material. Thus, the ACU server (i.e., GCKS) in proposed system provides complete isolation to every user account. Detailed functional description of GDOI is available in [26] and is out of scope for this paper.

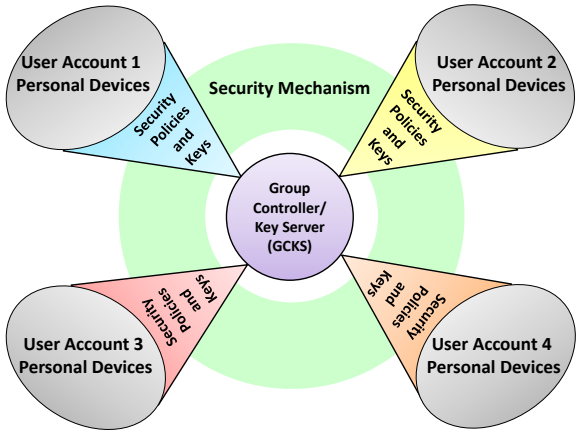


Figure 7: The GDOI-based communication security mechanism. All devices belonging to a single user have same security policies and keying material. However, every user has different security policies and keying material due to GDOI grouping feature.

4. Home Gateway Proxy

People keep their computers powered ON 24/7 due to two main reasons: (i) Internet connectivity for remote desktop/access and (ii) Internet connectivity for applications. The HGP deals with first issue while intelligent coordination among devices (addressed in Section 3) deals with the second issue. Home gateway devices are ideal location for proxy functionalities as they are often powered ON 24/7. The HGP impersonates only basic networking protocol (i.e., ARP and PING) on behalf of sleeping devices and wake them up on new connection attempt (e.g., remote desktop on TCP:3389 or UDP:3389, SSH remote access on TCP:22, Telnet remote access on TCP:23 etc).

4.1. Basic Architectural Design

The basic conceptual architecture of HGP is similar to NCP [6] and implements only a very basic subset of practically realizable features. The architectural design of HGP is depicted in Fig. 8 and consists of four building blocks: (i) network sockets, (ii) packet filters, (iii) packet processor and (iv) a basic set of proxying rules. The network sockets are used for communication with client devices (i.e., power managed computers) and to proxy presence on their behalf. They are used in UPnP-based communication with client devices. Further, network sockets are also created for sending/receiving PING and ARP packets on behalf of sleeping devices. The packet filters are used to sniff packets intended for sleeping devices based on their registered proxying rules. The captured packets are forwarded to packet processor block which analyzes them and determines appropriate actions e.g., respond to packet, ignore it or wake-up sleeping device. The packet processor processes packets based on the instructions specified in the proxying rules. The HGP implements a proxying rule for each specific protocol that includes instructions on how to process such packets.

4.2. Proxying Rules

Unlike NCP, the HGP is very simple and needs to implement only three proxying rules:

1. *ARP Rule:* It provides instructions on how to process ARP packets on behalf of sleeping devices. This

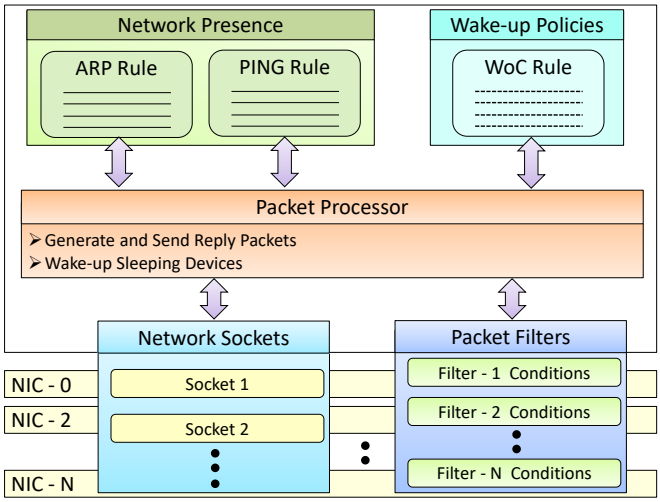


Figure 8: The basic architectural design of HGP.

rule is very important as it associates sleeping device IP address with the MAC address of HGP. Thus, all future packets intended for sleeping devices will be received by HGP.

2. *PING Rule:* It provides instructions on how to process PING packets on behalf of sleeping devices. PING requests are normally used for checking device presence/reachability. Thus, this rule enables HGP to impersonates presence of a sleeping device IP address in the network.
3. *Wake-on-Connection (WoC) Rule:* Originally specified in the NCP architecture [6], this rule provides instructions to wake-up a sleeping device when a packet intended for it contains a certain destination transport protocol and port number (e.g., remote desktop on TCP:3389 or UDP:3389, SSH remote access on TCP:22, etc).

4.3. Communication Framework

A communication framework is required to enable computers to communicate with the HGP. It is necessary for registration of proxying rules and notification of power state transitions. For hassle-free communication, the communication framework should inherit auto-discovery and seamless communication features. To this aim, multicast DNS (mDNS) or UPnP technology can be used. The mDNS is based on IP multicast over UDP protocol for device/service discovery. However, mDNS is designed for auto-discovery only and needs another protocol for communication [27]. Whereas, UPnP is a complete architecture that provides auto-discovery as well as seamless communication features between devices. Further, UPnP technology is most commonly used today by network devices and also easily portable on home gateways. Thus, the design of communication framework for HGP in our proposed system is based on UPnP technology.

The UPnP technology enables computers and HGP to automatically discover and communicate with each other

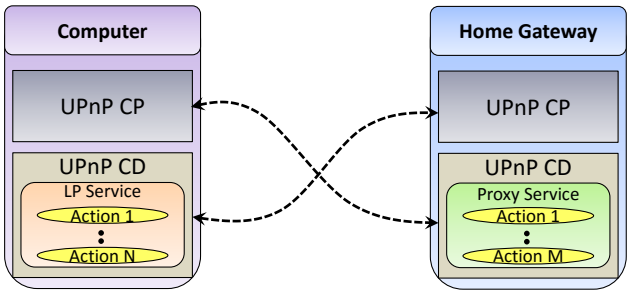


Figure 9: Design of UPnP communication framework.

as soon as connected to the network without needing any configurations or network settings. A UPnP device periodically advertises itself in the network and discovers and registers with other devices of interest. Detailed description of UPnP technology is available in [28] and is out of scope for this paper. Implementation instructions of the UPnP communication framework are provided in Section 5.

5. Implementation Guidelines and Tools

The implementation of proposed system (in Fig. 3) ensures hassle-free communication of user devices with the ACU server. All the functionalities in proposed system smoothly and seamlessly take place in background without requiring any user-settings or manual instructions. The ACU server transfers applications control from one device to the other autonomously. Further, a user don't need to inform manually its device operational state to ACU server, instead, a kernel module was developed that tracks power state transitions on computers and automatically informs the ACU server.

The proposed system consists of four software prototypes: (i) ACU server, (ii) ACU client, (iii) HGP and (iv) HGP client. The ACU client software runs on all user devices and communicates with ACU server for applications controlling based on priorities in current implementations. Based on the available devices and their priorities, the ACU server runs applications on the device with highest priority. The smartphone priority is normally set the lowest and it runs applications only if no desktop PC or laptop is running them. The HGP client software runs only on desktop PCs and laptops and communicates with HGP software running on the gateway device. The HGP client receives instructions from kernel module about power state transitions and immediately informs the HGP software over the UPnP communication framework. Whenever a computer is going to sleep state and wakes up, the power state change notification is sent to HGP (through HGP client) as well as to ACU server (through ACU client). The power state tracking is very important for both, HGP and ACU server to function correctly.

The HGP activates and de-activates proxying for its client devices based on their operational status. To get

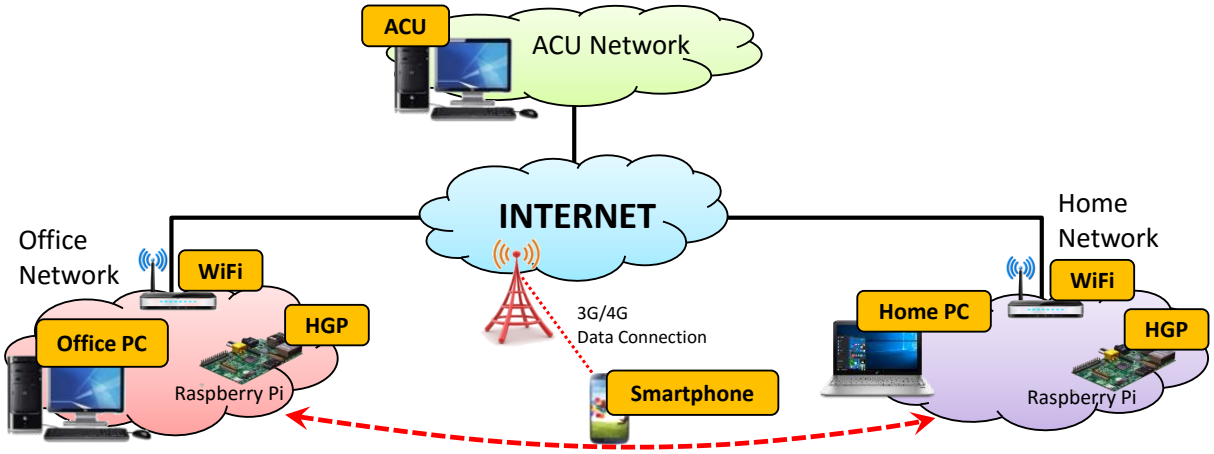


Figure 10: Experimental Testbed.

access to packets intended for its sleeping client devices, it implements traffic diversion based on ARP spoofing. The packet sniffing and filtering feature was implemented using Packet CAPturing (PCAP) libraries. Once a relevant packet is captured, it is forwarded to packet processor in HGP which then executes the appropriate action e.g., generates response, ignores it or wakes-up client device.

The proposed system requires two communication frameworks: (i) GDOI based secure HTTP and (ii) UPnP. The GDOI based secure HTTP communication framework is used for communication between ACU server and ACU client. Whereas, the UPnP communication framework is used for communication between HGP and its client devices. The authentication phase of GDOI security mechanism was implemented based on Diffie Hellman exchange [26]. The Diffie Hellman exchange enables ACU client to establish initial secret key with ACU server using public key cryptography techniques. Afterwards, the GDOI security mechanism periodically refresh security policies (e.g., encryption and signature algorithms etc) and keying material. Current implementations support AES-128-GCM, AES-256-GCM and AES-256-CBC encryption algorithms and HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 and AES-GMAC-128 signature algorithms. For ease in implementation of security algorithms, pyCrypto libraries were used.

The UPnP communication framework provides auto-discovery and zero-configuration based seamless communication. It consists of two main roles: controlled device (CD) and control point (CP). The CP functionalities are similar to a client that sends requests to a server. Whereas, the CD functions similar to a server. In standard UPnP communication system, one device implements only a CP and other implements only a CD. However, to enable two-way command and control features in proposed system, both the HGP and its client devices implement a CP as well as a CD (as shown in Fig. 9). The CD on HGP implements the proxy service that provides actions for ARP, PING and WoC rules registration. The CD on HGP client devices implements Low Power (LP) service that receives commands from the CP of HGP for power management of the device. The HGP can instruct its client devices about when to sleep. The LP service uses internal system calls to put the device into sleep state at the specified time. The UPnP communication framework was implemented using open-source libUPnP libraries. The implementations are compliant with UPnP v1.0 device architecture and successfully works on different OSes (Linux and Microsoft Windows).

All four software entities (ACU server, ACU client, HGP and HGP client) were implemented and tested in Linux operating system. The ACU server and ACU client were implemented using Python programming language with the help of pyCrypto libraries. Whereas, the HGP and HGP client were implemented using standard C/C++ programming language with the help of BOOST, PCAP and libUPnP libraries. Most of the networking tasks were implemented using BOOST libraries.

6. Measurements and Performance Evaluation

This section evaluates the proposed system for correctness of functionalities and presents performance metrics in real networking environments. The experimental testbed is depicted in Fig. 10 and consists of an office PC, home PC, a smartphone, an ACU server and two HGPs in home and office networks. Both, the home and office PCs run ACU client as well as HGP client softwares. However, the smartphone runs only the ACU client software. Since the ACU client software was developed for Linux operating system, it was tested on a Ubuntu based smartphone for experimentation purpose. The HGP software was executed on very low power Raspberry Pi pocket PC to avoid any incremental energy waste. As depicted in Fig. 10, the smartphone can be in the home network, in the office network or any where else using 3G/4G data connection.

First, we verified all the functionalities offered by developed software prototypes in proposed system. We noticed

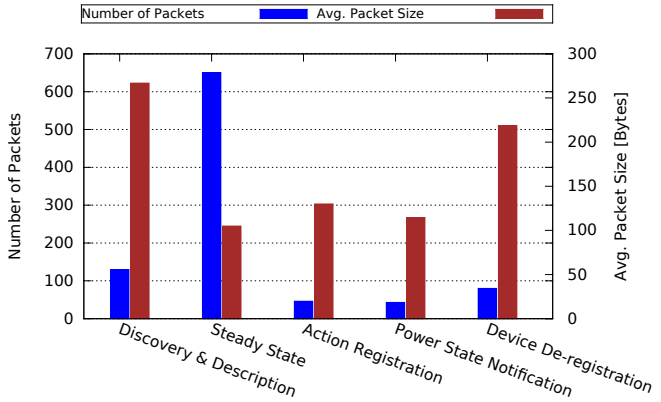


Figure 11: Number of packets exchanged and average packet size during different UPnP events.

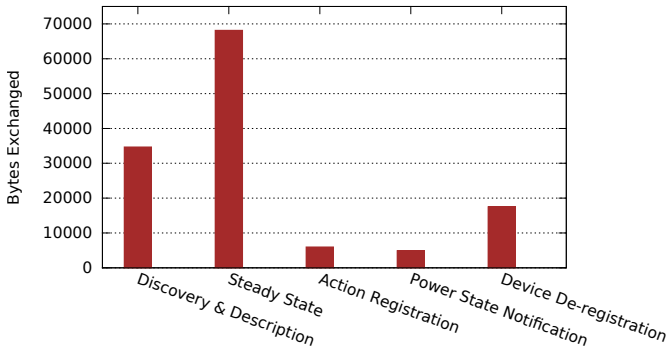


Figure 12: Total Bytes exchanged during different UPnP events.

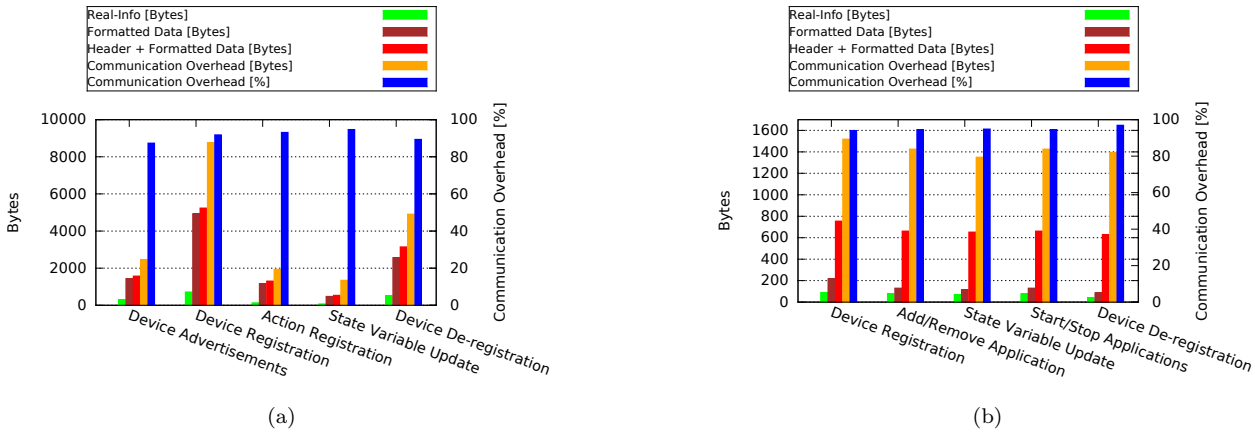
that ACU client and server softwares were correctly functioning. Applications controlling in current implementations are based on only devices priorities. The ACU server correctly executed applications on highest priority device and stopped on all other devices. When the highest priority device entered into sleep state, applications automatically started on the next high priority device without requiring any input or command from user. The whole process took place seamlessly. The kernel module was quite efficient in promptly detecting power state transitions and notifying the ACU server over GDOI based secure HTTP communication framework. The UPnP communication framework was also very efficient in providing auto-discovery between HGP and its client devices. The client devices seamlessly discovered HGP and communicated with it. The HGP client was very efficient in immediately transferring power state notifications (i.e., received from kernel module) to the HGP. Further, the HGP was able to timely wake-up the sleeping device and establish remote desktop/access connection. This feature is quite useful when a user wants to establish remote desktop connection with his sleeping computer in office from home or vice versa.

Next, we measured several performance critical factors in proposed system including communication overhead, latencies and resource requirements.

6.1. Communication Overhead

Communication overhead is a critical factor that can severely impact real-time performance for any application. Throughput is normally reduced for high communication overhead on low bandwidth channels. Further, packet loss may occur on low bandwidth channels if the communication framework has very high overhead. To analyze the overhead for our designed UPnP communication framework, we performed an experiment lasting 12 minutes during which HGP client discovers and registers with HGP, registers an action, changes its power state, stays in steady-state condition and finally de-registers with HGP. Fig. 11 depicts total number of packets exchanged during each individual UPnP event. It can be observed that most of the packets were exchanged during steady-state condition which are the periodic advertisement packets. It can also be observed in Fig. 11 that packets in steady state condition are very small in size. The average packet size during discovery/registration and de-registration is quite big as both of these phases involve downloading of complete UPnP device and service descriptions which are expressed in XML format. However, these big size packets are not frequent and only exchanged once during discovery/registration and de-registration. Fig. 12 provides more clear understanding by showing total number of Bytes exchanged during different events. Total Bytes exchanged are directly linked with packet transmission rate as well as its average size.

Fig. 13(a) presents dissection of packets during different UPnP events. The analysis is performed in terms of real information inside each packet, overhead due to structuring of real-information in XML format, overhead by also including headers and the total overhead due to UPnP communication semantics (i.e., a number of packets exchanged during each event). Real information means the actual data (e.g., IP & MAC address, device identifier, service identifier, state variables, action and its parameters etc) that needs to be transferred from one device to the other. UPnP represents this data in XML format before transmitting. It can be observed in Fig. 13(a) that real information inside UPnP packets is very small. Most additional overhead Bytes are due to XML formatting and UPnP communication paradigm. The device registration and de-registration phases have the highest overhead due to carrying complete device and server descriptions expressed in XML format. The overhead Bytes are less in action registration and state variable update due to carrying very small size of real information. Fig. 13(a) also depicts overhead Bytes percentage during different UPnP events. The overhead is more than 80% compared to real information during all UPnP events. However, the most frequent UPnP packets are very small in size while big size



(a) (b)

Figure 13: Overhead analysis: (a) UPnP communication overhead between HGP and its clients. (b) HTTP communication overhead between ACU client and ACU server.

packets are very infrequent. Thus, the high overhead does not leave adverse impact on UPnP communication.

Similar to UPnP communication framework, Fig. 13(b) presents dissection of packets exchanged between ACU client and ACU server. During registration, ACU client registers with ACU server. The ACU client can add or remove applications locally and also informs ACU server. The ACU server in turn informs all registered client devices. State variable update means notification about changes in device priority or power state. Based on priority, the ACU server instructs client device to start/stop applications. It can be observed in Fig. 13(b) that HTTP communication overhead is also quite high but the packets are very small compared to UPnP communication (in Fig. 13(a)). Due to small size packets, the overhead does not negatively impact communication allowing ACU clients and server to communicate smoothly.

6.2. Latencies Analysis

Another performance critical factor is latency that can severely affect operations on an application. First, we analyzed latencies as sum of HGP internal processing latency and UPnP communication latency in rules/actions registrations. The results are averaged over 100 tests and depicted in Fig. 14. It can be observed that latencies are independent of the specific rule. Further, latencies are very stable and small enough that it cannot negatively impact the HGP operations.

Since, HGP is processing packets on behalf of sleeping devices, the processing latency in generating response packets must be very small. We analyzed this for PING request packets in Fig. 15 considering: (i) HGP client device is awake and directly replying to packets and (ii) HGP client is sleeping and packets are responded by HGP on its behalf. It can be observed in Fig. 15 that HGP is quite efficient in replying to packets and average latency experienced by third-party device is lower than 2 ms. Further, no packet loss was observed during different trials.

The communication between ACU client and server is passing over the Internet. Therefore, communication latency will depend heavily on the available data rate and channel bandwidth. Further, HTTP is a simple client-server architecture that provides very low communication latency considering the available data rates today. Instead, we analyzed latencies due to GDOI security mechanism that protects HTTP communication against integrity and confidentiality attacks. Fig. 16 depicts the latencies for supported encryption algorithms with varying message size. The observed decryption latency is always slightly higher than the encryption latency. Further, latency increases slightly with the increase in message size. Similarly, Fig. 17 depicts the latencies for supported signature algorithms with varying message size. It can be observed that even for considerable message size of 10,000 Bytes, the latencies of encryption and signature algorithms are lower than 10 ms. Thus, GDOI security mechanism is very efficient and does not impact real-time communication between ACU client and server.

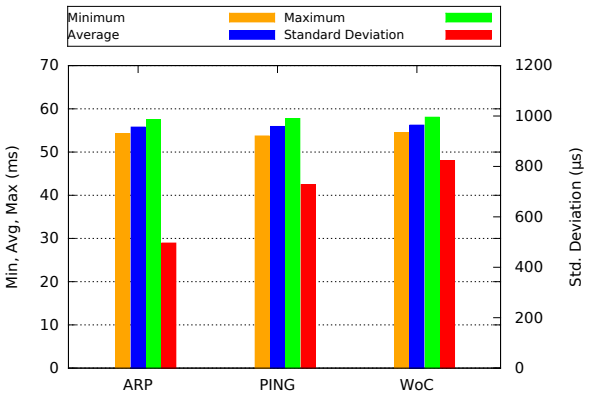


Figure 14: Delay experienced in registration of rules with HGP.

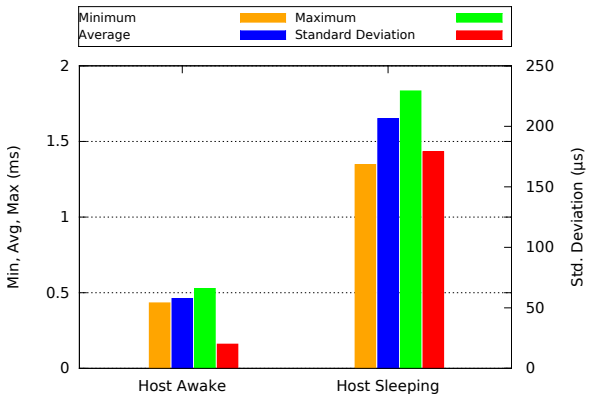


Figure 15: Round trip time for PING requests when responded by the host itself (i.e., awake) and by the HGP (i.e., host sleeping). The results are based on 100 PING packets.

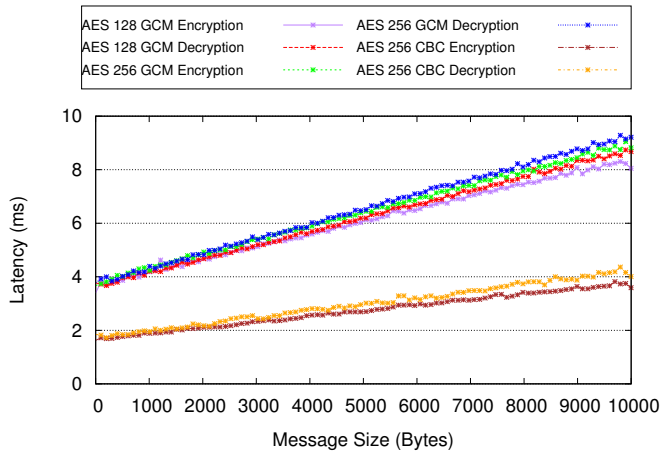


Figure 16: Processing latencies introduced due to supported encryption algorithms.

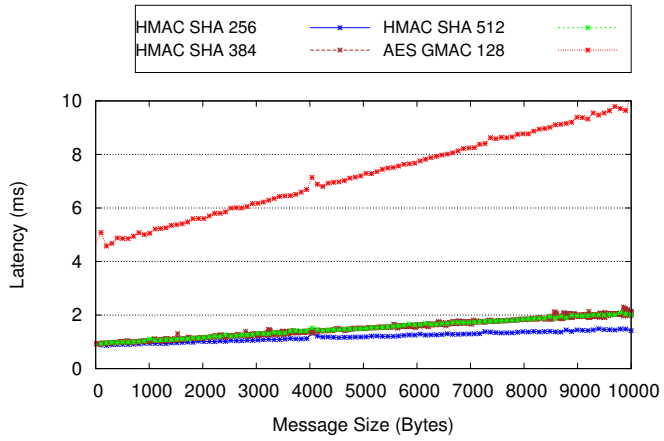


Figure 17: Processing latencies introduced due to supported signature algorithms.

Table 1: Memory requirements by developed software prototypes.

ACU Client	ACU Server	HGP Client	HGP
1.491 MB	1.798 MB	2.105 MB	2.67 MB

Table 2: Additional required memory per client device for HGP. The results are averaged over 100 clients.

Min (KB)	Avg (KB)	Max (KB)	Mean Dev (KB)
14.93	18.071	21.143	0.869

6.3. Memory Requirements

We analyzed memory and CPU usage by all four developed software prototypes (ACU client, ACU server, HGP and HGP client). The CPU usage was negligible due to no complex or CPU intensive functionalities involved. The memory usage can be critical especially for HGP which is expected to run on legacy gateway device. The gateway devices are normally equipped with 16 MB or 32 MB of memory limiting the possibilities to run additional softwares. Memory requirement is not critical for HGP client, ACU client and ACU server. The HGP client is expected to run on desktop PCs and laptops where as ACU client is expected to run on desktop PCs, laptops and smartphones as well. Today, these devices are equipped with several Gigabytes of memory extending their scope to virtually every kind of big size and complex applications. Further, memory requirement is also not a challenging task for ACU server. Table 1 presents memory requirements for all developed software prototypes. It can be observed that the memory requirements are very low. The HGP software requires only 2.67 MB of memory with no client device registered. Further, registration of a single client device requires only 18 KB of additional memory on average as reported in Table 2. This means that a legacy gateway device with only 8 MB of free memory can enable HGP to impersonate presence for up to 450 client devices. This is a quite big number as HGP scope is limited to only local network with very few client devices.

6.4. Expected Energy Savings

The proposed system reduces energy waste due to fixed network devices (e.g., desktop PCs) as well as helps mobile devices (e.g., laptops, tablets, smartphones) to significantly improve their battery life. The energy savings are achieved by allowing devices to sleep when idle without losing the network connectivity. The savings achievable from mobile devices will be negligible as they are already quite low powered devices. However, the proposed system can benefit them to improve their battery life e.g., smartphones are not running applications 24/7 in proposed system. Note, the applications on smartphones consume significant portion of battery due to utilizing hardware components such as CPU, memory, Wi-Fi, 3G/4G or built-in sensors. For desktop PCs and laptops, the energy savings heavily depend on their idle and sleep power consumptions as well as the amount of time devices can sleep. A

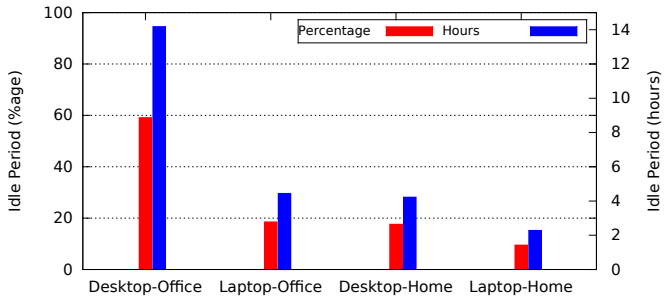


Figure 18: Daily idle duration estimated for devices from traffic traces in office and home environments [1].

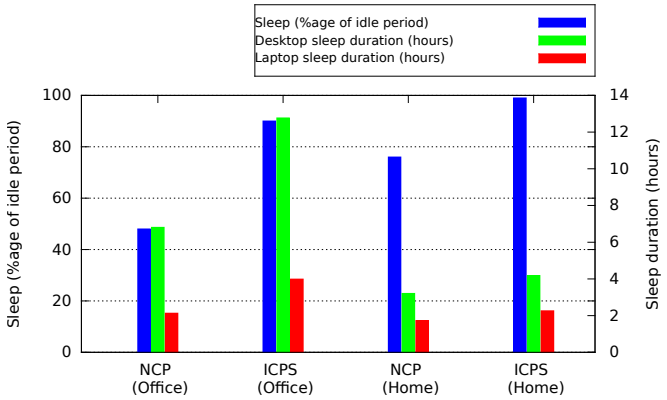


Figure 19: Daily sleep duration estimated for devices in office and home environments.

device sleep duration indirectly also depends on the HGP capabilities.

It is reported in [13] that about 30% of a typical smartphone battery is consumed by Internet connected applications. The rest of battery is consumed by operating system during its idle state. In proposed system, smartphones run applications only for short durations when no other device is running them (e.g., outside work hours). Further, the smartphones can be instructed not to run applications during user sleeping hours (e.g., 10:00 pm until 8:00 am). We expect battery usage by applications on smartphone lower than 10% in the proposed system. Assuming a typical smartphone battery on full charge lasts for 24 hours if running applications 24/7. Its battery duration can be increased up to 30.8 hours in the proposed system. This means at least 23% increase in the smartphone battery life.

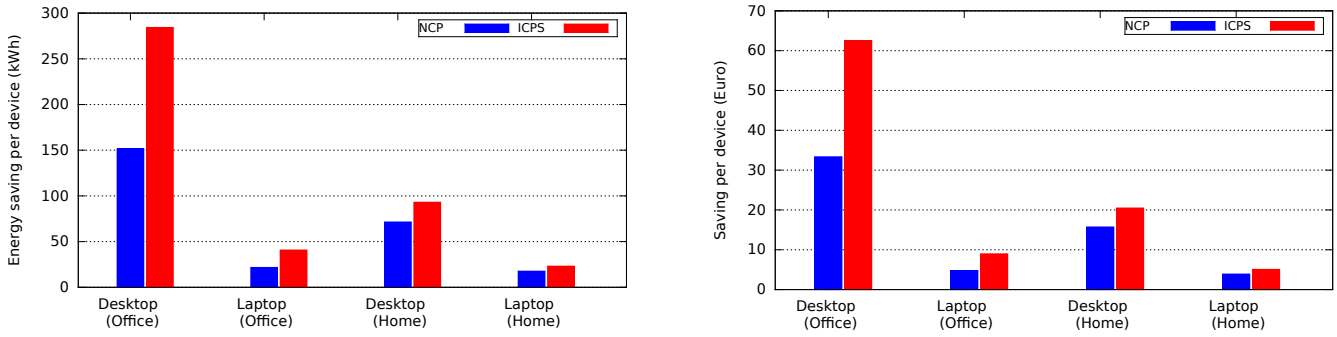
Analysis of network traffic is required to determine how long computers can sleep in office and home environments. Authors in [1] analyzed traffic samples to determine what type of applications and protocols are most frequently used by computers and which of them are proxiable. Based on results in [1], Fig. 18 depicts estimated daily idle duration for desktop PCs and laptops in office and home environments. It is worth to mention that this section assumes the NCP with only Basic Proxying (BP) capabilities i.e., it can proxy only simple network protocols (e.g., ARP, DHCP, PING, IGMP, NetBIOS etc) and wake up a client device whenever necessary. Majority of daily used applications are proprietary closed-source which cannot be proxied by NCP. The proposed system, Intelligent Collaborative Proxying Scheme (ICPS) can provide Full Proxying (FP) capabilities by allowing computers to sleep and maintaining their applications presence on smartphones. Authors in [1] specified that computers can sleep for 48% and 76% of the idle duration with BP (i.e., NCP) in office and home environments, respectively. However, computers can sleeping for 90% and 99% of idle duration with FP (i.e., the proposed ICPS system) in office and home environments, respectively. Based on these results, Fig. 19 depicts estimated sleep duration for desktop computers and laptops in the NCP concept and in the proposed ICPS system. It can be observed that the proposed ICPS system significantly increases sleep duration for devices when compared to the NCP concept.

To estimate the potential annual energy savings, we assumed 30 W and 2 W power consumptions for a typical laptop in idle and sleep state, respectively. For desktop computers, we assumed 65 W and 4 W power consumptions in idle and sleep state, respectively. Based on the daily sleep duration of computers and their idle and sleep power consumptions, Fig. 20(a) depicts the potential annual energy savings for a single computer in office and home environments. It can be observed that the proposed ICPS system provides much higher energy savings than the NCP concept. Its mainly because the NCP cannot proxy daily used proprietary closed-source applications. Fig. 20(b) depicts that a single desktop computer can provide annual savings of up to 60 Euro considering 22 cents per kWh as average electricity price in Europe. If the proposed system is adopted worldwide, billions of Euro savings can be achieved considering millions of devices in the world.

7. Conclusion

The NCP concept seems quite promising in terms of reducing network energy waste. It allows network devices to sleep without losing their presence over Internet. However, the NCP concept still faces several issues concerning proxying of proprietary closed-source applications with encrypted packets and TCP sessions. All proposed NCP strategies in literature are limited to open-source applications or propose changes to original applications and standard TCP/IP.

To overcome the limitations of NCP concept, this paper has presented an intelligent coordination scheme among daily used devices for reducing network energy waste. The



(a) Savings in kWh

(b) Savings in Euro

Figure 20: Expected annual savings per device.

proposed system is based on the fact that today we are using multiple devices concurrently, all having similar capabilities. Thus, a smartphone can maintain presence of applications when the desktop computer is sleeping and vice versa. The proposed system also relies on a lightweight HGP with very basic set of practically realizable features. The HGP maintains presence of sleeping computers and wakes them up on new connection attempts (e.g., a user accessing his sleeping computer in office from home and vice versa). The paper proposed UPnP as idle choice for the design of communication framework for HGP. It enables HGP and its client devices to automatically discover and seamlessly communicate with each other without requiring network configurations or user input. For protection of communication from cyber attacks, the paper proposed GDOI as an effective mechanism. Its periodic security policies and keying material refreshment mechanism provides strong protection against cryptanalysis.

The paper also presented experimental results by evaluating developed software prototypes in real network environment. The performance metrics validated the successful functionalities in proposed system including the GDOI security mechanism and developed communication frameworks. Further, the paper estimated that proposed system can provide equivalent or higher energy savings compared to the NCP concept.

At present, software prototypes were developed and analyzed in Linux OS. Future work will focus on extending support for other operating systems as well as performing detailed experimental analysis in real-world situations.

References

[1] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, N. Taft, Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 381–394.

[2] K. J. Christensen, C. Gunaratne, B. Nordman, A. D. George, The Next Frontier for Communications Networks: Power Management, Comput. Commun. 27 (18) (2004) 1758–1770. doi:10.1016/j.comcom.2004.06.012.

[3] R. Khan, R. Bolla, M. Repetto, R. Bruschi, M. Giribaldi, Smart Proxying for Reducing Network Energy Consumption, in: Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on, 2012, pp. 1–8.

[4] R. Khan, S. U. Khan, Achieving Energy Saving through Proxying Applications on behalf of Idle Devices, Procedia Computer Science 83 (2016) 187 – 194, the 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops. doi:http://dx.doi.org/10.1016/j.procs.2016.04.115.

[5] M. Jimeno, K. Christensen, B. Nordman, A Network Connection Proxy to Enable Hosts to Sleep and Save Energy, in: 2008 IEEE International Performance, Computing and Communications Conference, 2008, pp. 101–110. doi:10.1109/PCCC.2008.4745133.

[6] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Network Connectivity Proxy: Architecture, Implementation and Performance Analysis, IEEE Systems Journal PP (99) (2015) 1–12. doi:10.1109/JSYST.2015.2438639.

[7] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Network Connectivity Proxy: An Optimal Strategy for Reducing Energy Waste in Network Edge Devices, in: Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on, 2013, pp. 1–6. doi:10.1109/TIWDC.2013.6664214.

[8] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Design and Implementation of Cooperative Network Connectivity Proxy Using Universal Plug and Play, in: The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons, 2013, pp. 323–335.

[9] L. Irish, K. J. Christensen, A Green TCP/IP to Reduce Electricity Consumed by Computers, in: Southeastcon '98. Proceedings. IEEE, 1998, pp. 302–305. doi:10.1109/SECON.1998.673356.

[10] C. Gunaratne, K. Christensen, B. Nordman, Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed, Int. J. Netw. Manag. 15 (5) (2005) 297–310. doi:10.1002/nem.565.
URL http://dx.doi.org/10.1002/nem.565

[11] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, R. Gupta, Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 365–380.

[12] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-

Y. Shae, C. Waters, A Study of Internet Instant Messaging and Chat Protocols, IEEE Network 20 (4) (2006) 16–21. doi:10.1109/MNET.2006.1668399.

[13] R. Bolla, R. Khan, X. Parra, M. Repetto, Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications, in: 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, 2014, pp. 123–130. doi:10.1109/NGMAST.2014.10.

[14] ProxZZZy for Sleeping Hosts, in: Standard ECMA-393. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-393.pdf, June 2012.

[15] R. Bolla, R. Khan, M. Repetto, Assessing the Potential for Saving Energy by Impersonating Idle Networked Devices, IEEE Journal on Selected Areas in Communications 34 (5) (2016) 1676–1689. doi:10.1109/JSAC.2016.2545414.

[16] M. Jimeno, K. Christensen, A Prototype Power Management Proxy for Gnutella Peer-to-Peer File Sharing, in: 32nd IEEE Conference on Local Computer Networks (LCN 2007), 2007, pp. 210–212. doi:10.1109/LCN.2007.93.

[17] P. Werstein, W. Vossen, A Low-Power Proxy to Allow Unattended Jabber Clients to Sleep, in: 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, pp. 390–395. doi:10.1109/PDCAT.2008.18.

[18] J. Klamra, M. Olsson, K. Christensen, B. Nordman, Design and Implementation of a Power Management Proxy for Universal Plug and Play, in: Proceedings of SNCW, 2005.

[19] Y. Agarwal, S. Savage, R. Gupta, SleepServer: A Software-only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 22–22.

[20] R. Bolla, M. Chiappero, R. Khan, M. Repetto, Saving Energy by Delegating Network Activity to Home Gateways, IEEE Transactions on Consumer Electronics 61 (4) (2015) 445–453. doi:10.1109/TCE.2015.7389798.

[21] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Smart Proxying: An Optimal Strategy for Improving Battery Life of Mobile Devices, in: Green Computing Conference (IGCC), 2013 International, 2013, pp. 1–6. doi:10.1109/IGCC.2013.6604478.

[22] M. Gundlach, S. Doster, H. Yan, D. K. Lowenthal, S. A. Watterson, S. Chandra, Dynamic, power-aware scheduling for mobile clients using a transparent proxy, in: Parallel Processing, 2004. ICPP 2004. International Conference on, 2004, pp. 557–565 vol.1. doi:10.1109/ICPP.2004.1327966.

[23] I. Kelnyi, . Ludnyi, J. K. Nurminen, Using home routers as proxies for energy-efficient bittorrent downloads to mobile phones, IEEE Communications Magazine 49 (6) (2011) 142–147. doi:10.1109/MCOM.2011.5783999.

[24] V. Looga, Y. Xiao, Z. Ou, A. Yl-Jski, Exploiting traffic scheduling mechanisms to reduce transmission cost on mobile devices, in: 2012 IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 1766–1770. doi:10.1109/WCNC.2012.6214070.

[25] J. Hare, D. Agrawal, A. Mishra, S. Banerjee, A. Akella, A network-assisted system for energy efficiency in mobile devices, in: 2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011), 2011, pp. 1–10. doi:10.1109/COMSNETS.2011.5716500.

[26] B. Weis, S. Rowles, T. Hardjono, The Group Domain of Interpretation (GDOI), in: Internet Engineering Task Force (IETF) Request For Comments (RFC): 6407, Oct. 2011.

[27] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Design of home energy gateway boosting the development of smart grid applications at home, in: 2013 4th Annual International Conference on Energy Aware Computing Systems and Applications (ICEAC), 2013, pp. 103–108. doi:10.1109/ICEAC.2013.6737646.

[28] UPnP forum, 2012. URL: http://www.upnp.org.

Rafiullah Khan received his B.Sc. degree in Electrical Engineering from University of Engineering and Technology Peshawar, Pakistan, master degree in Satellite Navigation and Related Applications from Politecnico Di Torino, Italy, and Ph.D. degree jointly from University of Genoa, Italy and Polytechnic University of Catalonia, Spain, in 2009, 2010 and 2014, respectively. He completed his Ph.D. under a joint degree Erasmus Mundus program, funded by the European Commission.

He is currently a Postdoctoral Research Fellow at Queen's University Belfast in United Kingdom. He has co-authored over 30 scientific publications in international journals and conference proceedings and carried our research activities in framework of several national and European research projects. His research interests include Ad-Hoc Networking, Green Networking and Cyber Security.

Sarmad Ullah Khan graduated in Electrical Engineering from University of Engineering and Technology, Peshawar, Pakistan in 2007. From 2007 to 2008, he was lecturer at CECOS university. In 2013, he received his PhD in Electronics and Telecommunication Engineering from the University of Politecnico di Torino, Italy. He is currently an Assistant Professor in CECOS university, Peshawar, Pakistan and Member of IEEE. His research interests include security in wireless sensor networks, Internet of Things, intelligent transportation system and content centric networking.